

Fabric shader



Overview

This shader attempts to recreate the look of fabric materials. This is a material shader and should be used with the **VRayOSLMtl** material. The shader is provided by Derek Flood.

The shader code is available here: [fabric.zip](#)

Shader code

The shader code is below.

fabric.osl

```
/*
 *
 * fabric.osl shader by Derek Flood (c)2015
 * from http://docs.sharktacos.com/misc.php?action=help&hid=66
 *
 */

float fresnelReflectionFactor(normal bumped_normal, float ior)
{
    float c = abs(dot(I, bumped_normal));
    float g = ior * ior - 1.0 + c * c;
    if (g > 0.0) {
        g = sqrt (g);
        float A = (g - c) / (g + c);
        float B = (c * (g + c) - 1.0) / (c * (g - c) + 1.0);
        return 0.5 * A * A * (1.0 + B * B);
    }

    return 1.0;
}

normal getBumpedNormal(color centerColor, color uColor, color vColor, float inverseBumpAmount)
{
    vector worldTangent = normalize(dPdu);
    vector worldBitangent = normalize(dPdv);
    vector worldNormal = normalize(N);

    vector average = vector(0.3333333);
    float center = dot (average, vector(centerColor));
    float ddu = center - dot(average, vector(uColor));
    float ddv = center - dot(average, vector(vColor));
```

```

    return normalize(ddu * worldTangent + ddv * worldBitangent + inverseBumpAmount * worldNormal);
}

surface fabric
(
    [[ string description = "artist based fabric material" ]]
    (
        /* Diffuse section */
        color Sheen_color = color(1.0, 1.0, 1.0) [[ string description = "Sheen color can be black for nylon sheen." ]],
        float Sheen_opacity = 1.0 [[ string description = "Determines amount of sheen effect as fibres pick up light at incident angle." ]],
        float Sheen_tint = 0.5 [[ string description = "Tints both the sheen and specular." ]],
        int Sheen_ramp_type = 3
        [[ string widget = "mapper",
            string description = "Six types of interpolation for sheen. 1-3 (linear, exponential and smooth) are sharper, 4-6 (Sigmoid, square root and Fresnel) are softer.",
            string options = "linear:1|exponential:2|Smooth:3|Sigmoid:4|Square_root:5|Fresnel:6" ]],

        string Diffuse_color = "",
        float Diffuse_weight = 0.8,
        float Diffuse_roughness = 0.4,
        /* Spec section */
        color Spec_color = 1.0,
        float Spec_amount = 0.5,
        float Spec_glossiness = 0.6,
        float IOR = 1.33 [[ string description = "Determines the strength of Fresnel reflections; fabric generally has low frontal reflections." ]],
        int Subdivs = 8,
        int Trace_reflections = 1
        [[ string widget = "checkBox" ]],
        int Fresnel = 1
        [[ string widget = "checkBox" ]],

        /* Anisotropy section */
        float Anisotropy = 0.25,          // Fabric is anisotropic due to weave.
        float Aniso_rotation = 0,

        /* Bump section */
        string Bump_texture = "bump.png",
        float Bump_amount = 1.0,
        output color result = 1 )
    {
        /* Define Bump */
        normal bumped_normal = N;
        if (Bump_amount > 0.0)
        {
            float delta = 0.004;
            color center = texture(Bump_texture, u, v);
            color uColor = texture(Bump_texture, u + delta, v);
            color vColor = texture(Bump_texture, u, v + delta);

            bumped_normal = getBumpedNormal(center, uColor, vColor, 1.0 / Bump_amount);
        }

        /* Define Main color */
        color MainColor = texture (Diffuse_color, u, v, "missingcolor", color(1,0,0));
        MainColor = pow (MainColor, 2.2);

        /* Define Edge color */
        color TintBoost = transformc("hsl", MainColor);
        TintBoost[2] = pow (TintBoost[2], 0.2);
        TintBoost[2] = clamp(TintBoost[2],0,1);
        TintBoost = transformc("hsl","rgb", TintBoost);

        color EdgeColor = Sheen_color / 10;
        color TintEdge = EdgeColor * TintBoost;
        EdgeColor = mix (EdgeColor, TintEdge, Sheen_tint);
        EdgeColor = clamp (EdgeColor, 0, 1);
    }
}

```

```

/* Define Spec color */
color SpecColor = Spec_color;
color TintedSpec = SpecColor * TintBoost;
SpecColor = mix (SpecColor, TintedSpec, Sheen_tint);

/* Define Ramp */

float facingRatio = 1 - abs(dot(I, bumped_normal));

if( Sheen_ramp_type == 1) // linear
{facingRatio = facingRatio; }
if( Sheen_ramp_type == 2) // exponential (Down)
{ facingRatio *= facingRatio; }
if( Sheen_ramp_type == 3) // smooth
{ facingRatio = smoothstep (0,1, facingRatio ); }

if( Sheen_ramp_type == 4) // sigmoid S-curve
{
float Sigmoid = facingRatio / (1 + abs(facingRatio));
facingRatio = clamp( 2 * Sigmoid, 0, 1);
}
if( Sheen_ramp_type == 5) // square root
{ facingRatio = sqrt (facingRatio); }

if( Sheen_ramp_type == 6) // fresnel
{ facingRatio = 2 * fresnelReflectionFactor(bumped_normal, IOR); }

/* Fresnel */
if (Fresnel)
{ SpecColor *= fresnelReflectionFactor(bumped_normal, IOR); }
if (Trace_reflections)
{ MainColor *= (1.0 - SpecColor); }

/* BRDF Mixes*/
color SheenMix = EdgeColor * facingRatio * Sheen_opacity;
color EdgeMask = mix(1, Sheen_color, (facingRatio * Sheen_opacity) );
MainColor *= EdgeMask;
MainColor *= (1.0 - SheenMix);

closure color sheen_component = SheenMix * diffuse(bumped_normal, "roughness", Diffuse_roughness);
closure color diffuse_component = MainColor * Diffuse_weight * diffuse(bumped_normal, "roughness",
Diffuse_roughness);
closure color specular_component = Spec_amount * SpecColor *
vray_blinn (bumped_normal, Spec_glossiness, Anisotropy, Aniso_rotation,
"subdivs", Subdivs, "trace_reflections", Trace_reflections);

closure color reflect_component = SpecColor * reflection(bumped_normal, IOR);
Ci = diffuse_component + specular_component + reflect_component + sheen_component;
}

```